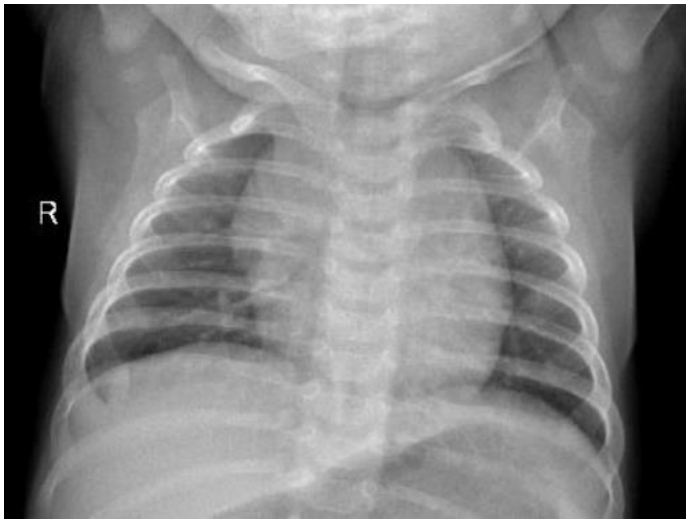
Task 2: Computer Vision using CNNs

Dataset: ***3 kinds of Pneumonia | Kaggle***

The task required me to implement a CNN that has a small-scale VGG architecture and also contains SENet blocks. i.e, Squeeze and Excitation blocks.

The dataset contains chest-xray images that will be fed into a CNN along with its corresponding label.



← this is what the data looks like

The process explained:

1. Data Collection
2. Data Preparation
3. Modelling
4. Evaluation

Data Collection:

Using kaggle's API, I downloaded the data into colab and then turned it into a dataframe which contains the path to each image as well as which class the image belongs to.

Then I split the data into training, testing and validation data. Training and validation data would be used to train the model while the testing data will be used to evaluate the model.

Data Preparation

Tensorflow provides an ImageDataGenerator class from which we can create an instance of it. This will basically take the path of the image from the dataframe and extract its data. Before the data is extracted, the ImageDataGenerator can also perform some data augmentation such as rotation of the image, zooming into the image, cropping it, etc.

Now the data is ready to be fed into a CNN.

Modelling

For this I created a CNN with VGG architecture that employs Squeeze and Excitation blocks.

VGG architecture:

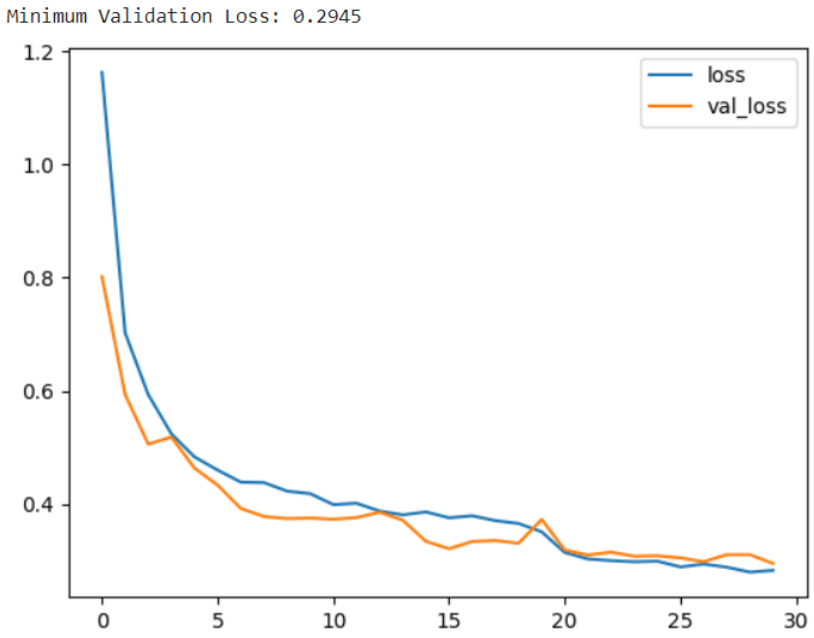
The entire model is divided into VGG_Blocks and each block has 1 convolutional layer followed by maxpooling and a dropout layer to increase performance.

There are about 7 such blocks where the dimensions of the data is gradually flattened. For example a 300 x 400 image is gradually made smaller as each convolutional layer extracts features of the image.

For the training phase, early_stopping as well as learning_rate_reduction was used to stop the training or reduce the learning rate if after a set number of epochs, no increase in the model's validation performance is noticed. This is used to prevent overfitting.

The training took about 5 hours to complete on google colab.

Evaluation



The above image shows how the loss and validation loss changed with each passing epoch.

	precision	recall	f1-score	support
0	0.96	0.99	0.97	292
1	0.80	0.86	0.83	266
2	0.72	0.59	0.65	145
3	0.99	0.96	0.97	113
accuracy			0.87	816
macro avg	0.87	0.85	0.86	816
weighted avg	0.87	0.87	0.87	816

This is the performance on the test dataset. The model is decently trained as it achieved a really good F1 score.